Templates

Create new theme/template

Keep in mind "MyName" is an example. You can name it anything you want.

1. Create lowercase directory: /tpl/myname
2. Copy file /tpl/default/tpl.php to /tpl/myname/tpl.php and edit the file.
   change `class Poodle_Output_TPL_MHX`
   into `class Poodle_Output_TPL_MyName`
3. Create file /tpl/myname/css/style.css for custom styling (optional)
4. Create file /tpl/myname/html/layouts/default.xml for custom html layout (optional)

TAL (Template Attribute Language)

The CMS uses a template system based on Zope's TAL system (wikipedia).
However, METAL is not supported.

More information can be found at:
Using Zope Page Templates
Advanced Page Templates
Appendix C: Zope Page Templates Reference

TAL Commands

TAL consists of seven different commands (highest priority first): define, condition, repeat, content, replace, attributes, and omit-tag.
Commands are attributes on HTML or XML tags, e.g. <div tal:content="article">Article goes here</div>

tal:define

Syntax: tal:define="[local | global] name expression [; define-expression...]</p> </p><p>Description: Sets the value of "name" to "expression". By default the name will be applicable in the "local" scope, which consists of this tag, and all other tags nested inside this tag. If the "global" keyword is used then this name will keep its value for the rest of the document.

Example: `<div tal:define="global title book/theTitle; local chapterTitle book/chapter/theTitle">`

tal:condition

Syntax: tal:condition="expression"

Description: If the expression evaluates to true then this tag and all its children will be output. If the expression evaluates to false then this tag and all its children will not be included in the output.

Example: `<h1 tal:condition="IDENTITY/isAdmin">Welcome admin to this page!</h1>`

tal:condition-else

Syntax: tal:condition-else="expression"

Description: This attribute is added in our system to construct faster if/elseif/else blocks. If the expression is omitted it is interpreted as the else block, else it is an elseif block. If the expression evaluates to true then this tag and all its children will be output. If the expression evaluates to false then this tag and all its children will not be included in the output.

Example: `<h1 tal:condition-else="IDENTITY/isMember">Welcome member to this page!</h1>`

tal:repeat

Syntax: tal:repeat="name expression"

Description: Evaluates "expression", and if it is a sequence, repeats this tag and all children once for each item in the sequence. The "name" will be set to the value of the item in the current iteration, and is also the name of the repeat variable. The repeat variable is accessible using the TAL path: repeat/name and has the following properties:

1. **index** - Iteration number starting from zero
2. **number** - Iteration number starting from one
3. **even** - True if this is an even iteration
4. **odd** - True if this is an odd iteration

5. **start** - True if this is the first item in the sequence
6. **end** - True if this is the last item in the sequence. For iterators this is never true
7. **length** - The length of the sequence. For iterators this is maxint as the length of an iterator is unknown
8. **letter** - The lower case letter for this iteration, starting at "a"
9. **Letter** - Upper case version of letter
10. **roman** - Iteration number in Roman numerals, starting at i
11. **Roman** - Upper case version of roman

Example:

```
<table>
<tr tal:repeat="fruit basket">
<td tal:content="repeat/fruit/number"></td>
<td tal:content="fruit/name"></td>
</tr>
</table>
```

tal:content

Syntax: tal:content="[text | structure] expression"

Description: Replaces the contents of the tag with the value of "expression". By default, or if the "text" keyword is present, the value of the expression will be escaped as required (i.e. characters "&<> will be escaped). If the "structure" keyword is present then the value will be output with no escaping performed.

Example: `<h1 tal:content="IDENTITY/nickname"></h1>`

tal:replace

Syntax: tal:replace="[text | structure] expression"

Description: Behaves identically to tal:content, except that the tag is removed from the output (as if tal:omit-tag had been used).

Example: `<h1>Welcome <b tal:replace="IDENTITY/nickname"></b></h1>`

tal:attributes

Syntax: tal:attributes="name expression[;attributes-expression]"

Description: Evaluates each "expression" and replaces the tag's attribute "name". If the expression evaluates to nothing then the attribute is removed from the tag. If the expression evaluates to default then the original tag's attribute is kept.

Example: `<a tal:attributes="href IDENTITY/website;title IDENTITY/nickname">Your Homepage</a>`

tal:omit-tag

Syntax: tal:omit-tag="expression"

Description: Removes the tag (leaving the tags content) if the expression evaluates to true. If expression is empty then it is taken as true.

Example: `<h1><b tal:omit-tag="not:IDENTITY/isMember">Welcome</b> to this page!</h1>`

TALES Expressions

The expressions used in TAL are called TALES expressions. The simplest TALES expression is a path which references a value, e.g. RESOURCE/body references the body property of the resource object.

path

Syntax: [path:]string[|TALES Expression]

Description: A path, optionally starting with the modifier 'path:', references a property of an object. The '/' delimiter is used to end the name of an object and the start of the property name. Properties themselves may be objects that in turn have properties. The '|' ("or") character is used to find an alternative value to a path if the first path evaluates to 'Nothing' or does not exist.

Example: `<p tal:content="book/chapter/title | string:Untitled"></p>`

There are several built in paths that can be used in paths:

1. **nothing** - acts as NULL in PHP
2. **default** - keeps the existing value of the node (tag content or attribute value)
3. **repeat** - access the current repeat variable (see tal:repeat)

4. **attrs** - a dictionary of original attributes of the current tag

There are also several built in default context properties that can be used:

1. **KERNEL** - the \Poodle::getKernel() object
2. **root** - the \Poodle::getKernel() object
3. **IDENTITY** - the \Poodle::getKernel()->IDENTITY object
4. **user** - the \Poodle::getKernel()->IDENTITY object
5. **SERVER** - the $_SERVER array
6. **RESOURCE** - the \Poodle::getKernel()->RESOURCE object
7. **REQUEST** - returns an array('GET'=>$_GET,'POST'=>$_POST)
8. **CONFIG** - the \Poodle::getKernel()->CFG object
9. **URI_BASE** - the \Poodle::$URI_BASE string
10. **URI_MEDIA** - the \Poodle::$URI_MEDIA string
11. **SQL** - the \Poodle::getKernel()->SQL object

exists

Syntax: exists:*path*

Description: Returns true if the *path* exists, false otherwise. This is particularly useful for removing tags from output when the tags will have no content.

Example: `<h2 tal:condition="exists:book/chapter/title" tal:content="book/chapter/title"></h2>`

not

Syntax: not:tales-*path*

Description: Returns the inverse of the tales-path. If the path returns true, not:path will return false.

Example: `<p tal:condition="not:IDENTITY/isMember">Welcome anonymous to the site!</p>`

Example: `<p tal:condition="not:php:0 == ${IDENTITY/id}">Welcome anonymous to the site!</p>`

string

Syntax: string:*text*

Description: Evaluates to a literal string with value *text while substituting variables with the form ${pathName} and $pathName*

Example: `<b tal:content="string:Welcome ${IDENTITY/nickname}!"></b>`

Example: `<b tal:content="'Welcome ${IDENTITY/nickname}!'"></b>`

php

Syntax: php:*php-code*

Description: Evaluates the php-code and returns the result. The PHP code must be properly escaped, e.g. "php: 1 < 2" must be written as "php: 1 &lt; 2". The PHP code has access to many PHP functions, including the TALES path as ${path}

Example: `<div tal:condition="php: ${basket/items} &gt; 0">Checkout!</div>`